A DIVIDE-AND-CONQUER METHOD FOR 3D CAPACITANCE EXTRACTION

A Thesis

by

FANGQING YU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2004

Major Subject: Computer Engineering

A DIVIDE-AND-CONQUER METHOD FOR 3D CAPACITANCE EXTRACTION

A Thesis

by

FANGQING YU

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Weiping Shi (Chair of Committee)	Jiang Hu (Member)
Jianer Chen (Member)	Chanan Singh (Head of Department)

May 2004

Major Subject: Computer Engineering

ABSTRACT

A Divide-and-Conquer Method for 3D Capacitance Extraction. (May 2004)
Fangqing Yu, B.S., University of Electronics Science and Technology of China
Chair of Advisory Committee: Dr. Weiping Shi

This thesis describes a divide-and-conquer algorithm to improve the 3D boundary element method (BEM) for capacitance extraction. We divide large interconnect structures into small sections, set new boundary conditions using the *border* for each section, solve each section, and then combine the results to derive the capacitance. The target application is critical nets where 3D accuracy is required.

The new algorithm is a significant improvement over the traditional BEMs and their enhancements, such as the "window" method where conductors far away are dropped, and the "shield" method where conductors hidden behind other conductors are dropped. Experimental results show that our algorithm is 25 times faster than the traditional BEM and 5 times faster than the window+shield method, for medium to large structures. The error of the capacitance computed by the new algorithm is within 2% for self capacitance and 7% for coupling capacitance, compared with the results obtained by solving the entire system using BEM. Furthermore, our algorithms gives accurate distributed RC, where none of the previous 3D BEM algorithms and their enhancements can.

ACKNOWLEDGMENTS

First and foremost, thanks must go to Professor Weiping Shi, who has given me much advice and strong support.

I would like to acknowledge Jeremy L Lumpkin, who did a lot of work in parsing GDSII to polygon. And also, I would like to acknowledge Shu Yan and Xiang Lu. We have had many fruitful technical discussions.

And last but certainly not least, I want to thank my husband, Ruiyue Chen, for his love, patience and constant support.

This research was supported by NSF grants CCR-0098329, CCR-0113668, EIA-0223785, and ATP grant 512-0266-2001.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

As the feature size decreases and operating frequency increases, interconnect parasite has a growing impact on circuit performance. Therefore fast and accurate extraction algorithms are crucial to the timing verification of modern digital circuits. It is well known that 2D/2.5D capacitance extraction algorithms are fast but inaccurate [1]. For critical nets and clock trees, the industry require 3D algorithms that have high accuracy [3].

A. Problem Description

The capacitance of an m-conductor geometry is summarized by an $m \times m$ capacitance matrix $\mathbf{C}$. To determine the j-th column of the capacitance matrix, we compute the surface charges on each conductor produced by raising conductor j to unit potential while grounding the other conductors. Then C_{ij} is numerically equal to the charge on conductor i. This procedure is repeated m times to compute all columns of $\mathbf{C}$.

Most 3D capacitance extraction algorithms are based on the Boundary Element Method (BEM). Using BEM, each of the m potential problems can be solved using an equivalent free-space formulation where the conductor-dielectric interface is replaced by a charge layer of density σ. The charge layer satisfies the integral equation

$$\psi(x) = \int_{surfaces} \sigma(x') \frac{1}{4\pi\epsilon_0 \|x - x'\|} da', \tag{1.1}$$

where $\psi(x)$ is the known conductor surface potential, da' is the incremental conductor surface area, x, $x' \in \Re^3$, $x' \in da'$, and $\|x - x'\|$ is the Euclidean distance.

The journal model is *IEEE Transactions on Automatic Control.*

Galerkin scheme is often used to numerically solve (1.1) for σ. In this approach, the conductor surfaces are divided into n small panels, and on each panel A_i, a charge q_i is assumed uniformly distributed. Then an equation is written which relates the known potential on A_i, denoted by v_i, to the sum of the contribution of potential from charges on all n panels $A_1, A_2, \ldots, A_n$. The result is a dense linear system

$$\mathbf{Pq} = \mathbf{v}, \tag{1.2}$$

where $\mathbf{q} \in \Re^n$ is the vector of panel charges, $\mathbf{v} \in \Re^n$ is the vector of known panel potentials, and $\mathbf{P} \in \Re^{n \times n}$ is the potential coefficient matrix. Each entry of $\mathbf{P}$ is defined as

$$p_{ij} = \frac{1}{area(A_i)} \int_{A_j} \frac{1}{area(A_j)} \int_{A_j} \frac{1}{4\pi\epsilon_0 \|x_i - x_j\|} \, da_j da_i, \tag{1.3}$$

for panels A_i and A_j. The linear system (1.2) has to be solved to compute panel charges, and the capacitances are derived by summing the panel charges. Because $\mathbf{P}$ is dense and large, iterative methods, such as GMRES, are used for solving the equation.

B. Previous Research

Several fast algorithms have been proposed to solve (1.2), such as FastCap [11], the pre-corrected FFT algorithm [12], the singular value decomposition algorithm IES^3 [9], hierarchical refinement algorithm HiCap [13], multi-scale [14], geometry independent approximation [10], and others [2, 5]. Although these algorithms use the fact that charges far away can be approximated, the entire interconnect geometry is carried throughout the algorithm.

Beattie and Pileggi [4] analyzed the effect of dropping negative charged particles far away from a center positive charged particle, which is known as the "window"

method. However they did not consider continuous charged surface, or dropping positive charged particles. Bachtold, et al., [3] proposed to construct a tunnel around a critical nets to be solved by 3D BEM. They used the window effect and also the "shield" effect to construct the tunnel. However, their method will produce large error if the critical nets are simply divided.

C. Method Overview

In this thesis, we introduce a divide-and-conquer method – *DiCap* for 3D capacitance extraction. *DiCap* works as follow : it reads from standard industry layout format – GDSII, combines with corresponding technology file to produce a generic format interface with pin texts. Then goes to the divide-and-conquer algorithm for capacitance matrix calculation and distributed RC generating. The whole flow chart is shown in Fig. 1.

Divide-and-conquer is a strategy widely used in algorithm design. To make it possible for capacitance extraction, we invent the idea of *border*, which allows us to partition a long interconnect into small sections with little error. The border idea, together with the window and shield ideas, is implemented in a new divide-and-conquer algorithm. The algorithm divides a BEM problem into a number of independent BEM problems, solves each BEM problem, and combines the results to derive the solution for the original BEM problem. We also prove the capacitance matrix obtained by our algorithm is diagonally dominant. Furthermore, our divide-and-conquer algorithm is the first 3D algorithm that can produce distributed RC, which is important for timing verification.

The difference between the window method and our method [4] is that the window method only drops negative charged conductors, while our algorithm divides and

drops both positive and negative charged conductors and panels. This allows us to drop more conductors than the window method does. The improvement of our method over the tunnel method [3] is that the tunnel method does not divide the critical nets, while our algorithm does. Simulation results show our algorithm is 5 times faster than the tunnel algorithm, which uses window and shield, and the error produced by our algorithm is small.

D. Thesis Outline

In chapter II, we give all important algorithms of generating generic format from GDSII. In chapter III, we present the main ideas of divide-and-conquer, and give out the whole algorithm. We also prove the passivity. Experimental results are shown in Chapter IV and conclusions are drawn in Chapter V.

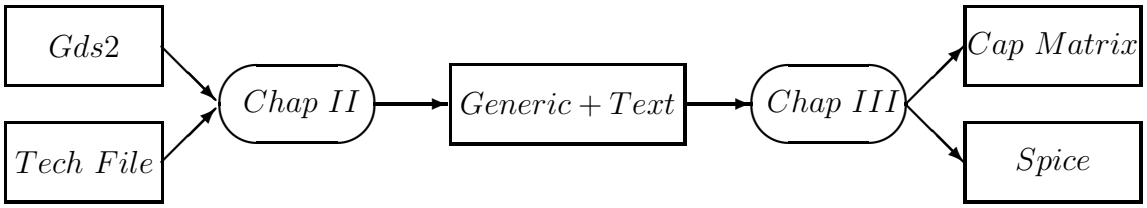

Fig. 1. DiCap extraction procedure.

CHAPTER II

GDSII TO GENERIC FILE INTERFACE

GDSII Stream format is the standard file format for transferring/archiving 2D graphical design data. It contains a hierarchy of structures, each structure containing elements (boundary/polygon, path/polyline, text, box, structure references, structure array references). The elements are situated on layers. It is a binary format. The GDSII format is a sequential list of records, each record contains a header to tell what information is the record. The order of the record needs to be according to the GDSII BNF (Bachus Nauer Forms). Most capacitance extraction program can read files in a simple generic format, such as FastCap [11] and [13]. This is a format that each panel's coordinates are listed in a single line along with its parent conductor name. In order to let these existing program to processor real layout, we need to transferring a gds2 to generic.

There are three major steps that are important in transferring a gds2 file into a generic interface, duplicate nested hierarchy structure, remove overlaps and subdivide polygons into rectangles. The details are presented in the following sections.

A. Duplicate Structure

Since GDSII contains a hierarchy of structures, which means one structure may be referenced by other structure. So it doesn't give out all information directly. We need to find the hidden information in the hierarchy structure. Fig. 2 is the algorithm to produce all coordinates and text information of the GDSII. The output of the algorithm is all the boundary/path locations , text and text locations of all structures. In step 2, the base structure means a structure that is not referenced by other structures. It is structure 0. One GDSII can have one or more than one base structure. The step

2 and 3 are for building data structure.

B. Subdivide Polygons

The interconnects are usually described as polygons in GDSII. But most capacitance extraction program can only take generic format as input. This is a rectangle based format. We need to subdivide GDSII polygons into rectangles. This is a relative more complex problem because the divided rectangles should be as square as possible. This needs more study. We currently only studied the most simple case, see Fig. 3. (a) is a polygon with six pairs of coordinates. After division, we can get two rectangles with each has four pairs of XY coordinates, see Fig. 3 (b) or (c).

The algorithm Subdiv-Poly (P) is to to divide a polygon shown in Fig 3 (a). It is shown in Fig. 4. It is assumed that all lines of the polygon are either parallel to X axis or parallel to Y axis.

C. Remove Overlaps

Most capacitance extraction programs can't handle overlaps. So we need to take out overlap areas of these rectangles. In general, there are three relationship between two rectangles: (1) no overlap (2) totally overlay (3) partially overlap, see Fig. 5. It is easy to handle the first two cases. For the first case, nothing needs to be done. For the second case, just remove the small one. The most complex case is the third one. We give the whole algorithm in Fig. 6. In step 7 of Fig. 6, we keep one of the rectangle, shrink the other one and create a new rectangle p to hold the rest, see Fig. 7.

Algorithm Dup-Stru (fp).

Input: fp is a GDSII file.

Output: All elements necessary for extraction.

Method:

1: Take all structures in.

2: Let the base structure(s) as root(s),

3: Let Structures refer to other structure as child of this structure.

4: Start with (one of) the root(s).

5: Get the children information layer by layer of the hierarchy tree and the offset of their srefs.

End of Procedure

Fig. 2. Duplicate structure algorithm.

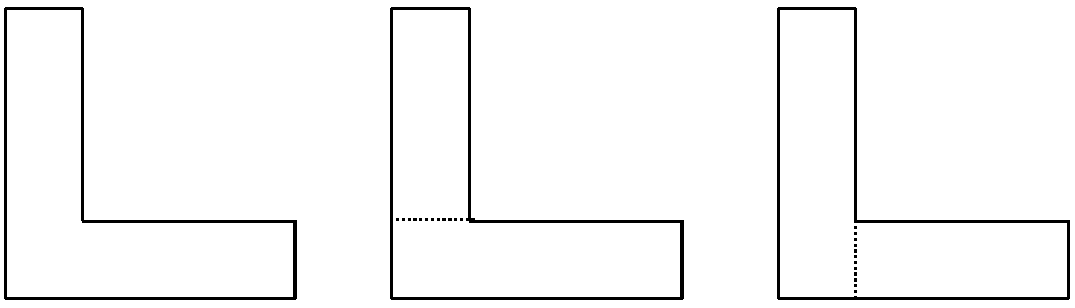

Fig. 3. (a) is a polygon needs subdivision, (b) and (c) are two possible cases after division.

Algorithm Subdiv-Poly (P).

Input: P is a input polygon.

Output: Rectangles corresponding to P.

Method:

1: **If** P has fours pairs of coordinates.

2: **Then** Output P.

3: **Else**

4: Let $X1$, $X2$, $Y1$, $Y2$ be the $Minx$, $Maxx$, $Miny$, $Maxy$ value of P.

5: Search the two points a, b with smallest X value $X1$.

6: Let their Y values be $y1$ and $y2$ ($y1 < y2$).

7: Output one rectangle with $Minx$, $Maxx$, $Miny$, $Maxy$ to be $X1$, $X2$, $y1$, $y2$ respectively.

8: Let the second X value of the polygon be xm.

9: **If** $y2 < Y2$

10: Output a rectangle with $Minx$, $Maxx$, $Miny$, $Maxy$ to be xm, $X2$, $y2$, $Y2$.

11: **Else**

12: Output a rectangle with $Minx$, $Maxx$, $Miny$, $Maxy$ to be xm, $X2$, $Y1$, $y1$.

End of Algorithm

Fig. 4. Subdivide a polygon into rectangles algorithm.

D. Connectivity Detection

GDSII doesn't provide any information about connectivity. But all capacitance extraction programs need the connected panels (rectangles) have the same conductor number, whatever they are in different layers. The algorithm of connectivity detection is shown in Fig. 8.

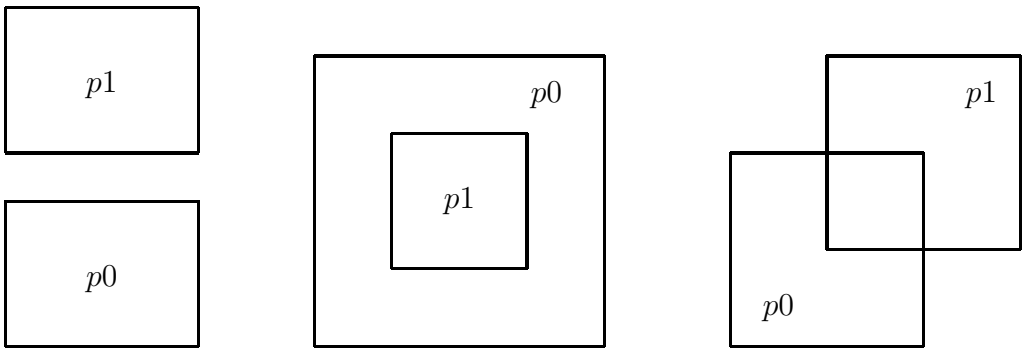

Fig. 5. (a) no overlap, (b)one rectangle totally overlap the other, (c) partially overlap.

Algorithm Remove-overlap (P).

Input: P is a linked list of rectangles with their coordinates.

Output: A vector of rectangles without overlap.

Method:

```
1: For each rectangle p0.
2:    For each rectangle p1.
3:       If there is overlap
4:       Then
5:          If one totally overlap the other
6:          Then remove the smaller one from the linked list P.
7:          If one partially overlap the other
8:          Then keep p0, change the coordinates of p1,
            create a new rectangle p to hold the rest, insert p
            the linked list P.
```

End of Algorithm

Fig. 6. Remove overlap algorithm.

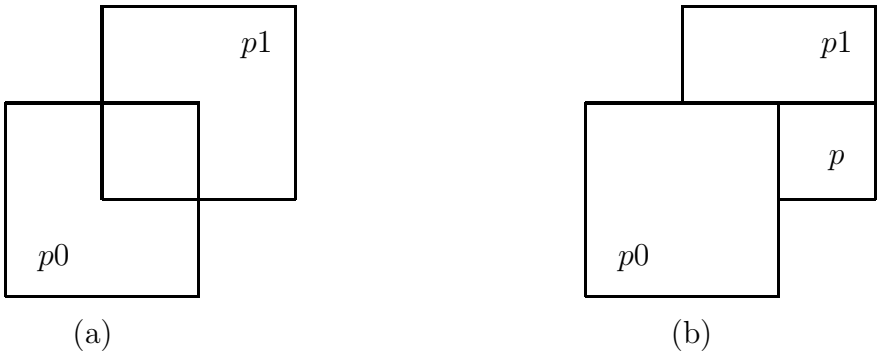

Fig. 7. (a) before remove, (b) after remove.

Algorithm Con-det (P).

Input: P is a vector of rectangles with their coordinates and layer number.

Output: A vector of rectangles with each group of connected rectangles have the same index.

Method:

1: **For** each rectangle $p0$.

2: **For** each rectangle $p1$.

3: **If** $p1$ is connected with $p0$

4: **Then** link $p1$ to the same linked list as $p0$, change all rectangles with the same index as $p1$ to the index of $p0$.

5: **ELSE** Create a new linked list with $p1$ as header, and give a new index.

End of Algorithm

Fig. 8. Connectivity detection algorithm.

CHAPTER III

ALGORITHM

In this chapter, we first discuss the main ideas of divide-and-conquer algorithm. In the second section of this chapter, the whole algorithm will be given. The last section of chapter is the simulation results.

A. Main Ideas

1. Divide-and-Conquer

Almost all BEM algorithms have superlinear time complexity in terms of the input size n, which is often the number of discretized panels. A superlinear function is one that grows faster than $O(n)$, such as $O(n \log n)$. The precorrected FFT algorithm [12] and SVD algorithm [9] both require $O(n \log n)$ time. Although the fast multipole algorithm [8] is claimed to be $O(n)$ time, there is a hidden assumption that the n particles are uniformly distributed. If the distribution is non-uniform, then $\Omega(n \log n)$ time is necessary to build the multipole hierarchy [7]. In addition, as the problem size increases, the number of GMRES iterations will slowly increase. All these contribute the superlinear time.

Consider the special case where the BEM algorithm runs in $cn \log n$ time for some constant factor c. If we divide the extraction problem into 2 sub-problems of each size $n/2$, then the time for solving the two sub-problem using the BEM algorithm will be

$$2 \cdot \left(c\frac{n}{2} \log \frac{n}{2} \right) = cn \log n - cn.$$

Therefore, if the overhead for dividing the problem and combining the results is less than cn, then the divide-and-conquer strategy will save time.

We can not estimate in theory how much the divide-and-conquer strategy reduce running time, since we do not know exactly the constant factors in the BEM and in dividing and combining. However, our experiments show that when n is large and the size of each section is appropriate, then the benefit of divide-and-conquer is significant.

2. Border

In order to divide a conductor, we will attach a border to the faces where the conductor is cut. For example in Figure 9, we attach a border at each end of s_i. The border has the same shape as the neighboring sections s_{i-1} and s_{i+1}, but is much smaller in size, If the potential problem we want to solve is for $\psi(s_i) = V$, which is 1 or 0, we will set the borders at the same potential as s_i. However the charges on the borders are not counted towards s_i when computing the capacitance.

Fig. 10 shows the experimental result of a conductor divided and calculated with different length of border. The total length of the conductor is 10 units, which is divided into 3 section of length 3, 3, and 4 units respectively. We can see that a small border reduces the error significantly.

3. Shield

We can drop a conductor if the conductor is hidden behind another conductor. This idea was previously used in [3], but we will give a theoretical estimation on the error. Consider the conductors in Fig. 11, where $\psi(S_1) = 1$ and $\psi(S_2) = \psi(S_3) = 0$.

The solution of linear system (1.2) gives

$$q_3 = q_2 \frac{p_{22}p_{13} - p_{23}p_{12}}{p_{33}p_{12} - p_{32}p_{13}}.$$

Let the distance between S_1 and S_2 be 1, and the distance between S_2 and S_3 be d.

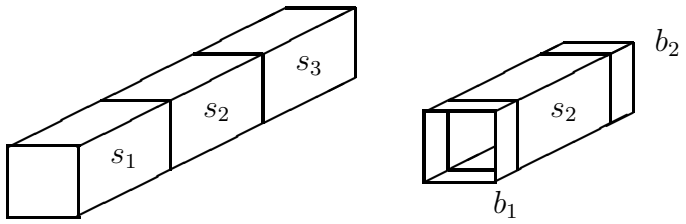

Fig. 9. $S = \{s_1, s_2, s_3\}$. To solve for s_2, attach border b_1 and b_2 and set $\psi(b_1) = \psi(b_2) = \psi(s_2)$.

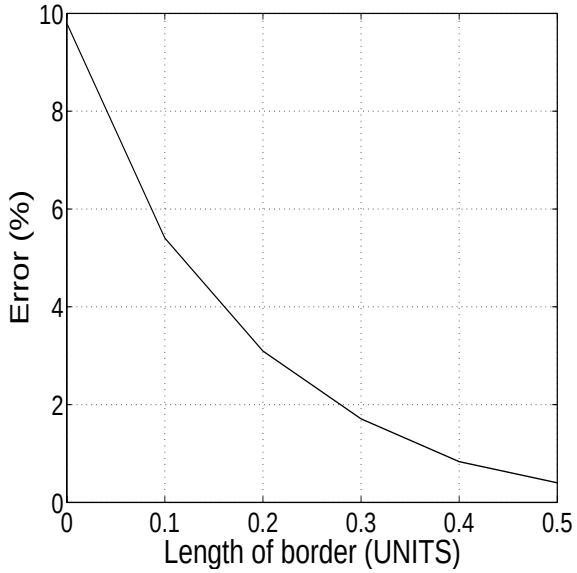

Fig. 10. A short border reduces error drastically.

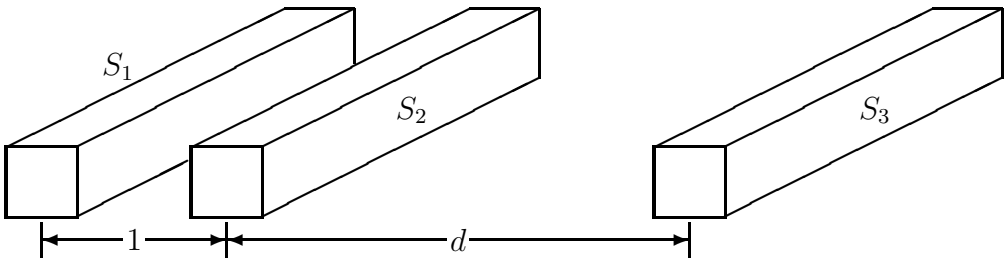

Fig. 11. Three conductors S_1, S_2 and S_3.

Assume S_2 and S_3 are the same size, therefore $p_{22} = p_{33}$. From (1.3), we know that p_{22} and p_{23} are much greater than 1, $p_{13} = p_{31} \approx 1/(1+d)$, and $p_{23} = p_{32} \approx 1/d$. Therefore,

$$q_3 \approx q_2 \frac{p_{22}\frac{1}{1+d} - \frac{1}{d}}{p_{33} - \frac{1}{d}\frac{1}{1+d}} = q_2 \cdot O(1/d).$$

Therefore, the potential contribution of S_3 to S_1 is about $p_{13}q_3 = q_2 \cdot O(1/d^2)$, which decays very fast as d increases.

Fig. 12 shows the error induced by dropping S_3 in a system with and without the shield S_2. It is obvious that dropping S_3 when it is behind a shield produces much less error than dropping S_3 when there is no shield.

B. Algorithm

We first give an overview of the algorithm. Then we describe how to divide a large system into small independent subsystems, followed by the main algorithm. Finally, we prove that our algorithm produces a capacitance matrix that is diagonally dominant.

1. Algorithm Overview

The divide-and-conquer algorithm works as follows. To solve the potential problem where the i-th conductor is at unit potential, we partition conductor i into several sections. For each section, we attach borders to the cutting faces, and set the borders at unit potential. We drop far away conductors and conductors that are hidden behind other conductors using a simple line-sweep algorithm. Then we solve the charge distribution for each section. Repeat the process for every section of the i-th conductor. Finally, we add up the charge on each section, and ignore the charge on the borders, to derive the total capacitance. To output distributed RC, the charge

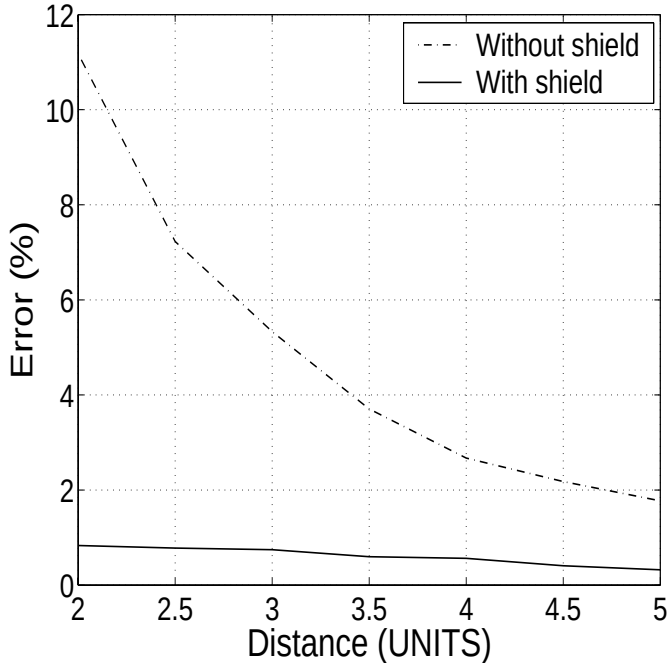

Fig. 12. As the distance between S_1 and S_3 change, the error of self-capacitance of S1 due to drop of S_3.

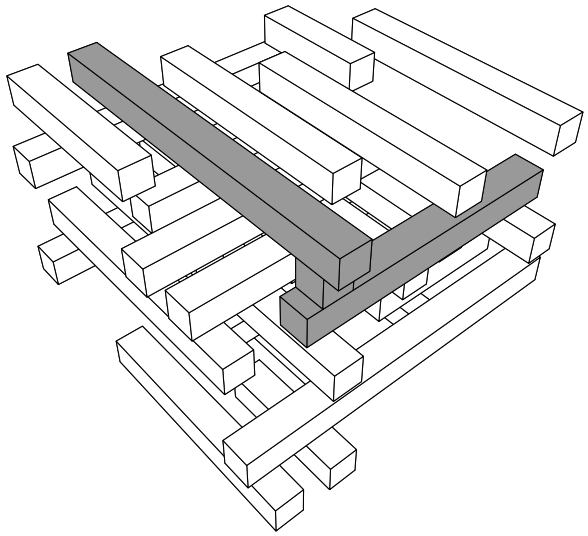

Fig. 13. A critical net in a 5-layer structure.

on each section is the distributed capacitance for that section. This way, we can compute the self capacitance and coupling capacitance between conductor i and its neighbors, either lumped or distributed. Figure 13 is a critical net to be extracted. Figure 14 is one section of the critical net with borders attached, and far away or hidden conductors dropped. For the sub-problem in Figure 14, we apply unit voltage on the dark and gray parts, and ground the other conductors.

2. Divide

Procedure Get-Window takes as input a conductor s, and returns a set of conductors in a window. Please notice, the output of this procedure is not just using distance property to drop conductors far away, but applying shield and border property to further drop unnecessary conductor for calculation.

Fig. 15 shows the 2D view of the window in the same layer as the target conductor s. There are three possible cases for a conductor s in a window as shown in Figure 15, where s is either completely inside of the window, or s extends outside of the window in one or both directions. A window contains regions A, B, N and conductor s. Region A is the *accumulation* region. Charges on conductors in A will be accumulated to compute the coupling capacitance. Region N is the *corner* region. Charges on conductors in N will be ignored. Region B is the *border* region and is on the same conductor with s. Charges on conductors in B will be ignored. When we solve the potential problem, conductors in B and s are set to unit potential, while all other conductors are grounded.

The procedure Get-Window is in Fig. 16, where window sizes W_x and W_y are shown in Fig. 15(a), $\min X(s)$ and $\max X(s)$ are the leftmost and rightmost X-coordinates of s, and $\min Y(s)$ and $\max Y(s)$ are the topmost and bottommost Y-coordinates of s.

In Step 9 and 11 Fig. 16, the visibility check is done by a line sweep approach [6]. Consider the case for the region above conductor s, assuming s goes in the X-direction, see Fig. 17. We first sort the endpoints of conductor panels in A and N in increasing X-coordinates. Then we scan the endpoints in increasing X order, to determine which conductor panel has the minimum Y coordinate. This line sweep algorithm can be done in linear time, together with the algorithm that builds the multipole hierarchy. Panels that are not visible from s or B are dropped.

Fig. 17(a) shows 4 conductors in the window above s and B. S_1 and S_2 are visible, S_3 is invisible, and S_4 is partially visible. Our visibility check drops S_3 and partitions S_4 into S_4' which is visible, and drops the rest of S_4 which is invisible. The final output for this example is $S_A = \{S_1', S_4'\}$, $S_N = \{S_1'', S_2\}$.

For the layers immediately above or below the current layer where the target conductor is located, all conductors inside of the window are kept. Fig. 13 shows the Get-Window being applied on a real case.

3. Conquer

After the charge on every conductor is solved by procedure Solve-Charge in Fig. 18, we can add the total charge to derive the row of capacitance matrix entries. Figure 19 is our main algorithm.

Note that in Step 7 of Fig. 19, we set conductor s_{ij} and its border S_B at unit potential, and ground the other conductors. This is the new boundary condition we created to reduce the singularity at the corner. In Step 8, we accumulate charges for conductor S_i, and in Step 9, we accumulate charges for conductors in S_A. Charges on conductors in S_B and S_N are ignored, because these conductors are used to set the boundary condition and may appear many times.

The three parameters in the algorithm, L, W_x and W_y, may affect the perfor-

mance. Window size W_y does not have significant impact on the accuracy and time since most neighbors are included anyway. We choose W_x as far as results meet accuracy requirement. The relationship between L and running time is complex: the greater (less) the L, the less (greater) the saving due to dividing, and the less (greater) the overhead.

C. Passivity

Now we prove the capacitance matrix computed by our algorithm is diagonally dominant. Previous passivity proof for the window method [4] is not enough since our algorithm also drops conductors with positive charge.

Lemma 1 *Given a set of small panels in 3D where each panel is set at potential 1 or 0. If we delete any panel of potential 1, then each remaining panel of potential 1 will contain more positive charge, and each remaining panel of potential 0 will contain less negative charge.*

Proof: First, let $\mathbf{P}$ be the coefficient matrix defined in (1.3), and let $\mathbf{P}^{-1} = \{b_{ij}\}$. Clearly, $\mathbf{P}^{-1}$ is the capacitance matrix of the panels. From properties of capacitance matrix, we know $\mathbf{P}^{-1}$ is diagonally dominant, with diagonal entries $b_{ii} > 0$ and off diagonal entries $b_{ij} < 0$, $i \neq j$.

Let the panel to be deleted be A_1. Let the potential of each panel A_i be v_i. Initially $v_1 = 1$. Now lower v_1 from 1 to x and consider the charge on every panel:

$$\begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} = \mathbf{P}^{-1} \begin{bmatrix} x \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} b_{11} \cdot x + \sum_{i=2}^{n} b_{1i} \cdot v_i \\ \vdots \\ b_{n1} \cdot x + \sum_{i=2}^{n} b_{ni} \cdot v_i \end{bmatrix}.$$

For any $j > 1$, $b_{j1} < 0$. Therefore for any $j > 1$,

$$q_j = b_{j1} \cdot x + \sum_{i=2}^{n} b_{ji} v_i.$$

Since the effect of removing A_1 is equivalent to lowering x to the background potential induced by potential on other conductors, x must be between 0 and 1. Therefore,

$$q_j > b_{j1} + \sum_{i=1}^{n} b_{ji} v_i.$$

Theorem 1 *The capacitance matrix* $\mathbf{C}$ *produced by the divide-and-conquer algorithm is diagonally dominant.*

Proof: Since conductors other than S contain negative charge, dropping those conductors will make the reduced system still positive definite [4]. Lemma 1 says that dropping the part of S outside the window will increase the amount of positive charge on s and decrease the amount of negative charge on S_A. Since s_{ij}'s do not overlap, nor S_A's overlap, the total positive charge on all s will be greater than the charge on S if we solve the whole system undivided. Similarly, the total negative charge on each conductor other than S will be less than that if we solve the whole system undivided. Since the capacitance matrix obtained by solving the whole system is diagonally dominant, the capacitance matrix obtained by using our algorithm will be even more diagonally dominant.

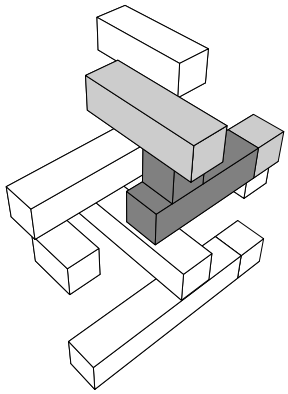

Fig. 14. One section of the critical net. The dark part is to be extracted, and the grey part is the border.

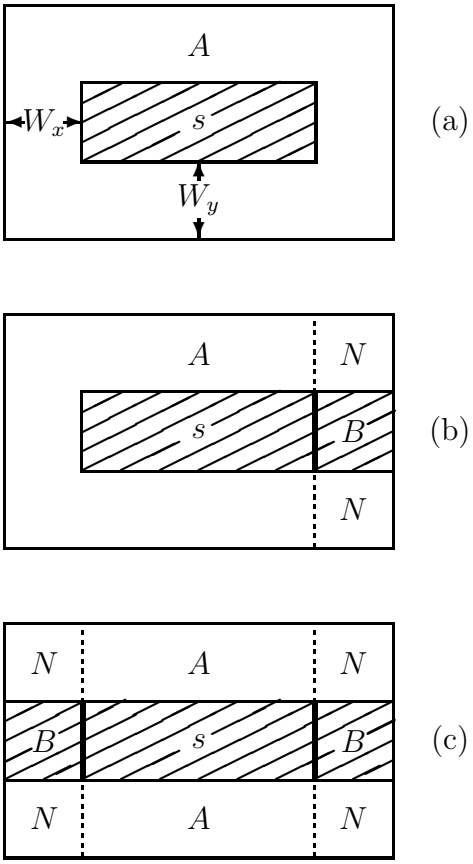

Fig. 15. Two-dimensional view of a window, where s is the target conductor, A is the accumulation region, B is the border, and N is the corner region. B and s belong to the same conductor.

Procedure Get-Window ($\mathbf{S}, S, s, W_x, W_y$).

Input: **S** is the set of all conductors;

S is the conductor at unit potential;

s is the section of S to be solved;

W_x and W_y are the size of the window.

Output: Sets of conductors S_A, S_B and S_N.

Method:

1: Let A be region $[\min X(s) - W_x, \max X(s) + W_x]$
$\times[\min Y(s) - W_y, \max Y(s) + W_y]$;

2: **If** $s = S$ **Then**

3: $B \leftarrow \emptyset$ and $N \leftarrow \emptyset$;

4: **Else**

5: Let B be the region shown in Fig. 15(c);

6: Let N be the region shown in Fig. 15(c);

7: $A \leftarrow A - B - N$;

8: **Endif**

9: $S_A \leftarrow \{$Conductors in A and visible from $s\}$;

10: $S_B \leftarrow \{$Conductors in B and on $S\}$;

11: $S_N \leftarrow \{$Conductors in N and visible from $B\}$;

12: **Return** S_A, S_B and S_N.

End of Procedure

Fig. 16. Get Window Procedure

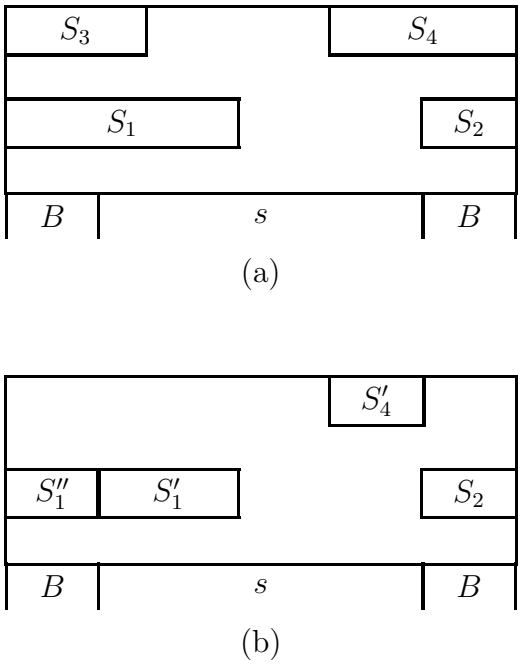

Fig. 17. In (a), four conductors are in the window. In (b), conductors invisible from s and B are deleted.

Procedure Solve-Charge (S_P, S_G).

Input: S_P is the set of conductors at unit potential,

and S_G is the set of conductors grounded.

Output: Charge vector Q for each conductor.

Method:

Any BEM.

End of Procedure

Fig. 18. Solve-Charge Procedure

Algorithm Divide-and-Conquer($\mathbf{S}, L, W_x, W_y$).

Input: A set of conductors $\mathbf{S}$ in 3D,

and user provided parameter L, W_x and W_y.

Output: Capacitance matrix $\mathbf{C}$.

Method:

For each conductor $S_i \in \mathbf{S}$ **Do**

  For each conductor $S_j \in \mathbf{S}$ **Do**

    $Q_j \leftarrow 0$;

  Divide S_i into disjoint sections $s_{i1}, \ldots, s_{in_i}$

  according to length L;

  For each section s_{ij} **Do**

    $(S_A, S_B, S_N) \leftarrow$ Get-Window($\mathbf{S}, S_i, s_{ij}, W_x, W_y$);

    $Q' \leftarrow$ Solve-Charge $(\{s_{ij}\} \cup S_B, S_A \cup S_N)$;

    Let $Q'(s_{ij})$ be the charge corresponding to s_{ij} in Q';

    $Q_i \leftarrow Q_i + Q'(s_{ij})$;

    For each $s_k \in S_A$, let $s_k \in S_m$ **Do**

      Let $Q'(s_k)$ be the charge corresponding to s_k in Q';

      $Q_m \leftarrow Q_m + Q'(s_k)$;

For each j **Do**

  $C_{ij} \leftarrow Q_j$.

End of Algorithm

Fig. 19. The main algorithm

CHAPTER IV

SIMULATION

A. Capacitance Matrix

The divide-and-conquer algorithm is implemented in C and the BEM algorithm is on HiCap [13]. The reason we chose HiCap is because it is easier to program and can solve larger structures than FastCap can. Nevertheless, we also experimented with FastCap, and found the speedup is similar to HiCap.

All computation are done on a SUN Ultra Enterprise 2. In all experiments, time is the total CPU time to compute one row of the capacitance matrix. The error of capacitance matrices is defined as follows. Let the row of capacitance matrix computed by direct BEM be $\mathbf{C}$ and the row of capacitance matrix computed by our algorithm be $\mathbf{C}'$. Then the error is estimated in the Frobenius norm: $\|\mathbf{C} - \mathbf{C}'\|/\|\mathbf{C}\|$. This is the standard way to measure the difference between two matrices. We also separate self capacitance with coupling capacitance, since the magnitude of self capacitance is much greater than the coupling capacitance.

The test examples are large 3-layer structures in Fig. 20 and results are in Tables I and II. We set the width and height of each conductor, minimum spacing and inter-layer dielectric thickness all at $0.5\mu m$. Three interconnects of length 50 μm, 100 μm, and $150\mu m$ respectively, are used for comparison. We first solve the whole system to compute the row of capacitance matrix for the long interconnect. The results are in the column "whole". Then we apply window+shield method to drop conductors far away or hidden from other conductors, see Fig. 21. Again we compute the row of capacitance matrix for the long interconnect, with the reduced system. The results are in the column "wind+shld". We divide the long interconnect into sections of 10

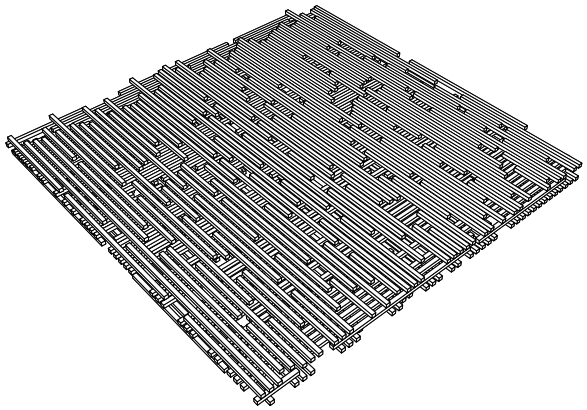

Fig. 20. A large dense 3-layer structure. The net to be extracted is running on 3 layers through 2 vias. The traditional method solves the whole system.

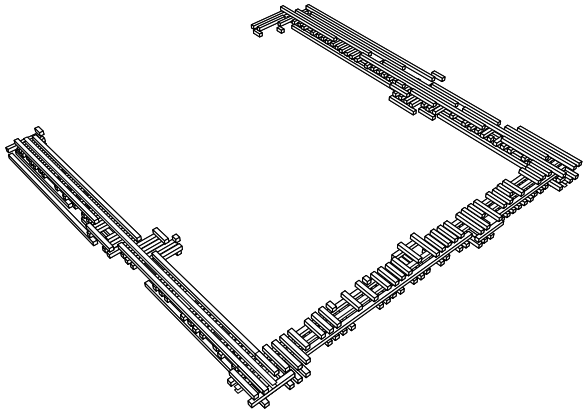

Fig. 21. The interconnect in Fig. 20 with its neighbors obtained by the window+shield method.

μm each, without adding the border. The results are in the column "wind+shld+div". Finally, we use the divide-and-conquer algorithm to partition the long interconnect into sections of 10 μm each with border, one is shown in Fig. 22. The results are in the column "new". As the length of each section decreases, more time is used for dividing and the overhead due to border also increases. We find that for using HiCap and our current divide-and-conquer implementation, the best choice for the length of each section is 10 μm, regardless of the length of the input interconnects. The relationship between time and length of each section in the 150 μm example is shown in Fig.23. The same optimal value for the length of each section is shown same in the 50 μm and 100 μm examples.

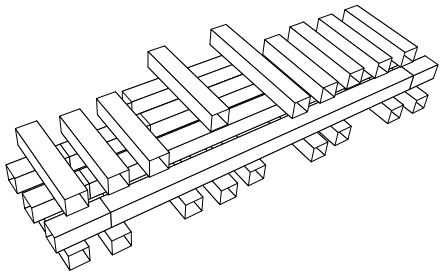

Fig. 22. One section of the interconnect in Fig. 20 and 21 with its neighbors. There are total 15 sections. The border can be seen.

Table I. Simulation results for long interconnects.

size	performance	whole	win +shld	win +shld +div	new
50 (μm)	time (sec)	755.7	8.8	8.1	8.4
	mem (MB)	391.5	15.8	2.7	2.9
	error (C_{11})	—	1.6%	8.7%	1.1%
	error (C_{1i})	—	6.7%	15.9%	6.7%
100 (μm)	time (sec)	755.7	90.5	20.6	21.8
	mem (MB)	391.5	41.3	2.7	2.9
	error (C_{11})	—	1.6%	8.8%	1.2%
	error (C_{1i})	—	6.5%	16.1%	6.4%
150 (μm)	time (sec)	755.7	152.7	28.8	30.0
	mem (MB)	391.5	56.8	2.7	2.9
	error (C_{11})	—	1.7%	8.7%	1.4%
	error (C_{1i})	—	6.8%	16.8%	6.8%

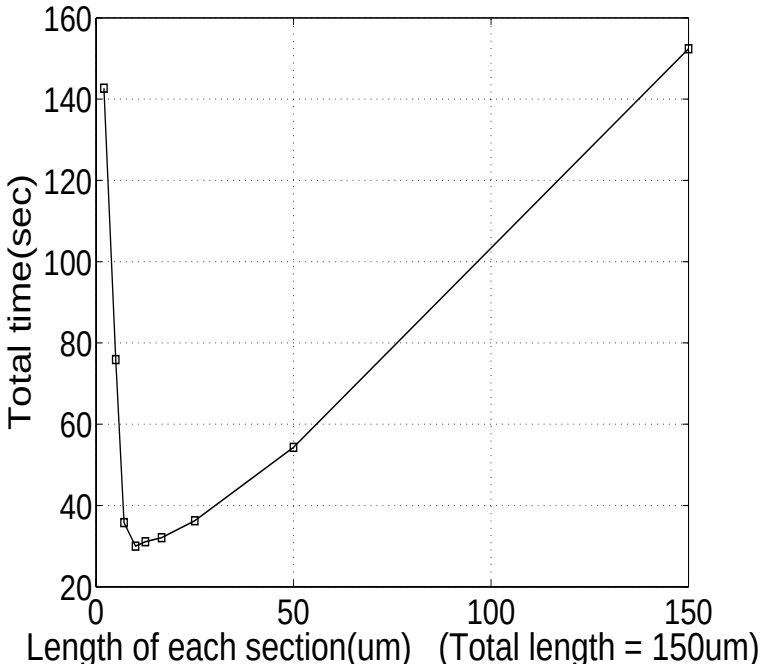

Fig. 23. Relationship between the length of each section and the CPU time.

Table II. Significant entries of $C_{1,i}$ (pF) for the large structure.

i	1	2	3	4	5	6	7
whole	**6939.66**	-289.60	-337.46	-321.93	-287.78	-280.45	-288.16
win+shld	**6821.69**	-286.25	-337.58	-319.58	-285.16	-283.40	-291.50
new	**6838.92**	-281.52	-341.16	-319.76	-278.43	-287.72	-293.81
i	8	9	10	11	12	13	14
whole	-168.78	-300.17	-329.80	-312.11	-266.45	-368.00	-40.71
win+shld	-171.93	-307.61	-326.21	-309.89	-263.55	-369.63	-44.73
new	-165.41	-307.23	-330.21	-309.39	-263.13	-369.50	-39.52

CHAPTER V

CONCLUSIONS

This thesis presents a divide-and-conquer method for 3D capacitance extraction. The method can take GDSII stream as input, and generating capacitance matrix and distributed RC automatically. This method is based on a divide-and-conquer algorithm that can significantly speed up BEMs without sacrificing accuracy. The algorithm is easy to implement and works for any BEM. The ideal application is for critical nets and clock tree where 3D accuracy is required. For the examples tested, the speed up is 25 and 5 times respectively compared with the traditional method that solves the whole system and the window+shield methods that drops far away and hidden conductors. Our memory usage is 1/130 to 1/20 of those used by the traditional method and the window+shield method. Compared with simply dividing without border, our result is much more accurate. Most of the error of our algorithm is induced by dropping negative part of the system rather than dividing a conductor. Dividing itself generate almost no error.

To apply a method to multi-layer dielectric, it is important that the method is kernel independent. Since our algorithm uses any BEM to solve the sub-problems, our algorithm is kernel independent as long as the BEM is kernel independent. There are many BEMs that are kernel independent, such as the SVD algorithm [9] and the hierarchical refinement algorithm HiCap [13].

REFERENCES

[1] N. D. Arora, K. V. Raol, R. Schumann and L. M. Richardson, "Modeling and extraction of interconnect capacitance for multilayer VLSI circuits," *IEEE Trans. CAD*, Vol. 15, No. 1, Jan. 1996, 58–67.

[2] M. Bachtold, M. Emmenegger, J. G. Korvink, and H. Baltes, "An error indicator and automatic adaptive meshing for electrostatic boundary element simulations," *IEEE Trans. CAD*, Vol. 16, No. 12, Dec. 1997, 1439–1446.

[3] M. Bachtold, M. Spasojevic, C. Lage and P. B. Ljung, "A system for full-chip and critical net parasitic extraction for ULSI interconnects using a fast 3-D field solver," *IEEE Trans. CAD*, Vol. 19, No. 3, Mar. 2000, 325–338.

[4] M. W. Beattie and L. T. Pileggi, "Error bounds for capacitance extraction via window techniques," *IEEE Trans. CAD*, Vol. 18, No. 3, Mar. 1999, 311–321.

[5] M. Beattie and L. Pileggi, "Electromagnetic parasitic extraction via a multipole method with hierarchical refinement," *Proc. 1999 ICCAD*, 437–444.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, New York, 1997.

[7] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *J. ACM*, Vol. 24, No. 1, Jan. 1995, 67-90.

[8] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems,* MIT Press, Cambridge, Massachusetts, 1988.

[9] S. Kapur and D. E. Long, "IES^3: A fast integral equation solver for efficient 3-dimensional extraction," *Proc. 1997 ICCAD*, 448–455.

[10] S. Kapur and D. E. Long, "Large-scale capacitance calculation," *Proc. 2000 DAC*, 744–749.

[11] K. Nabors and J. White, "FastCap: A multipole accelerated 3-D capacitance extraction program," *IEEE Trans. CAD*, Vol. 10, No. 11, Nov. 1991, 1447–1459.

[12] J. R. Phillips and J. White, "A precorrected FFT method for capacitance extraction of complicated 3-D structures," *IEEE Trans. CAD,* Vol. 16, No. 10, Oct. 1997, 1059–1072.

[13] W. Shi, J. Liu, N. Kakani and T. Yu, "A fast hierarchical algorithm for 3-D capacitance extraction," *IEEE Trans. CAD*, Vol. 21, No. 3, March 2002, 330–336.

[14] J. Tausch and J. White, "A multiscale method for fast capacitance extraction," *Proc. 1999 DAC*, 537–542.

VITA

Fangqing Yu's permanent address is Room 403, Taipingnan Road 114, Nanjing, Jiangsu,210002, P.R.China

Fangqing Yu received her bachelor's degree in June 1998 from the University of Electrical Science and Technology of China in electrical engineering. She received her degree in May 2004 from Texas A&M University, College Station in computer engineering.

The typist for this thesis was Fangqing Yu.